Evento: XXVII Jornada de Pesquisa

UTILIZAÇÃO DO ALGORITMO DE SHOR PARA QUEBRA DE CRIPTOGRAFIA RSA EM COMPUTADORES QUÂNTICOS [1]

BREAKING RSA CRYPTOGRAPHY USING SHOR'S ALGORITHM IN QUANTUM COMPUTERS

Evandro Fernandes de Uzeda[2], Ricardo Vanni Dallasen[3], Maikon Cismoski dos Santos[3], André Fernando Rollwagen[3], Daniel Delfini Ribeiro[3], José Antônio Oliveira de Figueiredo[3]

[1] Pesquisa desenvolvida no Instituto Federal de Educação Ciência e Tecnologia Sul Rio-Grandense, Câmpus Passo Fundo
[2] Bacharel em Ciência da Computação do IFSUL - Câmpus Passo Fundo
[3] Docente do Curso de Bacharelado em Ciência da Computação do IFSUL - Câmpus Passo Fundo

RESUMO

Este trabalho aborda a utilização de computação quântica para a quebra da criptografia RSA. Para isso é realizada a análise de como as chaves pública e privada são geradas a partir do produto obtido pela multiplicação entre dois números primos. Este algoritmo é baseado na dificuldade de se realizar a fatoração de um número grande para encontrar seus coeficientes, que são dois números primos. A computação quântica pode ser utilizada na etapa de fatoração dos primos de um produto para otimizar esse processo. Para realizar essa tarefa é utilizado o algoritmo de Shor. Para verificar a capacidade de efetuar a quebra de uma chave RSA, foi desenvolvido um programa utilizando o SDK do Qistik para realizar a quebra da criptografia RSA, executado em um simulador de computador quântico. As chaves utilizadas para os testes foram de valores pequenos devido a limitação de *qubits* do simulador.
Palavras-chave: Computação Quântica, Criptografia. RSA, Algoritmo de Shor.

ABSTRACT

This work addresses the use of quantum computing to break RSA encryption. An analysis is carried out on how the public and private keys are generated from the product obtained by multiplying two prime numbers. This algorithm is based on the difficulty of factoring a large number to find its coefficients, which are two prime numbers. Quantum computing can be used in the step of factoring the primes of a product to optimize this process. To accomplish this task, Shor's algorithm is used. To verify the ability to break an RSA key, a program was developed using the Qistik SDK to break the RSA encryption that runs in a quantum computer simulator. The keys used for the tests had small values due to the qubit limitation of the simulator.

Keywords: Quantum Computing. Cryptography. RSA. Shor's Algorithm

INTRODUÇÃO

O algoritmo recomendado para utilização em aplicações do tipo cliente-servidor é o RSA (Rivest-Shamir-Adleman), tendo em vista que a chave pública pode ser enviada aos clientes e somente o servidor tem posse da chave privada, para realizar a desencriptação das mensagens. Em conjunto com a chave pública, também é enviado o produto dos coeficientes que foram utilizados para gerar as chaves pública e privada. Isso não caracteriza necessariamente um problema, uma vez que para gerar as chaves ainda é necessário conhecer quais são os fatores primos utilizados para gerar o produto. É possível que um computador clássico encontre esses coeficientes e gere a chave privada, entretanto é necessário um número suficientemente grande de tentativas, o que torna o processo inviável. Este trabalho aborda a utilização de computação quântica para encontrar os fatores de um produto utilizando o algoritmo de Shor, com objetivo de melhorar o tempo de execução.

ALGORITMO DE CRIPTOGRAFIA RSA

O algoritmo RSA é baseado na dificuldade de se realizar a fatoração de um número grande para encontrar seus coeficientes, que são dois números primos . O produto obtido pela multiplicação entre dois números primos é utilizado para realizar a geração da chave pública e da privada. Essa multiplicação entre dois números primos é simples de ser calculada. No entanto, o processo reverso (partindo do produto, descobrir quais são os multiplicadores) é uma tarefa bastante dispendiosa computacionalmente. Dessa forma o RSA consegue manter seguro os números primos que são as bases da geração das chaves (MILANOV, 2009).

Para realizar a encriptação de um conteúdo é necessário possuir a chave pública. Para para desencriptar o conteúdo é preciso dispor da chave privada. Para o processo de geração das chaves é necessário obter dois números primos que sejam distantes um do outro e também de grande valor. Dessa forma, os números podem ser escolhidos utilizando o teste de primalidade, como o algoritmo de Rabin–Miller. Após encontrados os dois números primos, é realizado o cálculo do produto entre eles. O produto será utilizado no processo de encriptação da mensagem (MILANOV, 2009).

Geração dos números primos e seu produto

Para gerar as chaves é necessário obter dois números primos que sejam distantes um do outro e também de grande valor. Os números podem ser escolhidos utilizando um teste de primalidade, como por exemplo, usando o algoritmo de Rabin–Miller. Depois de encontrados os dois números primos que serão utilizados, é realizado o cálculo do produto entre eles. O produto será utilizado no processo de encriptação da mensagem (MILANOV, 2009).

Para calcular o produto pode ser utilizada a função de Carmichael. Essa função define o produto entre os dois números primos como sendo o valor do Mínimo Múltiplo Comum (MMC), com o decréscimo de uma unidade a cada número primo. A função é dada pela expressão

$$\lambda(n) = MMC((p - 1), (q - 1)) \quad (1)$$

onde n é o produto e p e q são os números primos respectivamente. O resultado da função é utilizado para realizar a desencriptação da mensagem (MILANOV, 2009). Também é possível utilizar o produto da multiplicação, que é dado como

$$\lambda(n) = (p - 1) * (q - 1) \quad (2)$$

Criação da chave pública

De posse de $\lambda(n)$, a chave pública é criada com a escolha de um número primo, chamado de e, que deve ser maior que 1 e co-primo de $\lambda(n)$. A encriptação da mensagem é realizada como descrito na equação

$$c = m^e \bmod n \quad (3)$$

onde m é a mensagem, n é o produto, e a chave pública e c é a mensagem criptografada.

Criação da chave privada

A fim de descriptografar a mensagem, é necessário saber o valor de d, que será a chave privada, sendo d o inverso de e tal como co-primo de $\lambda(n)$. Com isso, o último passo consiste em utilizar a equação

$$m = c^d \bmod n \quad (4)$$

onde m é a mensagem originalmente encriptada.

ALGORITMO DE SHOR

Como visto na seção anterior, o algoritmo RSA utiliza o produto de dois números primos grandes para gerar tanto chave pública quanto privada. O problema que o algoritmo de Shor resolve é: dado um número inteiro N é preciso encontrar outro número inteiro P que esteja no intervalo entre 1 e N e que seja divisor de N (IBM QUANTUM).

O algoritmo de Shor foi demonstrado utilizando um computador quântico com 7 qubits por um grupo da IBM em 2001, onde o número 15 foi fatorado nos números 3 e 5, (QUANTIKI, 2015).

O algoritmo de Shor pode ser dividido em duas partes. A primeira parte consiste em reduzir o problema de fatoração ao problema de encontrar o período de uma função, tarefa que pode ser realizada por um computador clássico. Na segunda parte é encontrado o período de uma função. Essa parte é realizada em um computador quântico (IBM QUANTUM).

Primeira parte

Para iniciar o processo de redução é necessário escolher um número a que seja menor que o produto N. Em seguida, é calculado o Maior Divisor Comum (MDC) entre a e N . Se o resultado do MDC for diferente de 1, essa etapa é concluída. Caso contrário, é necessário usar uma subrotina para encontrar o período r de uma função.

Segunda Parte

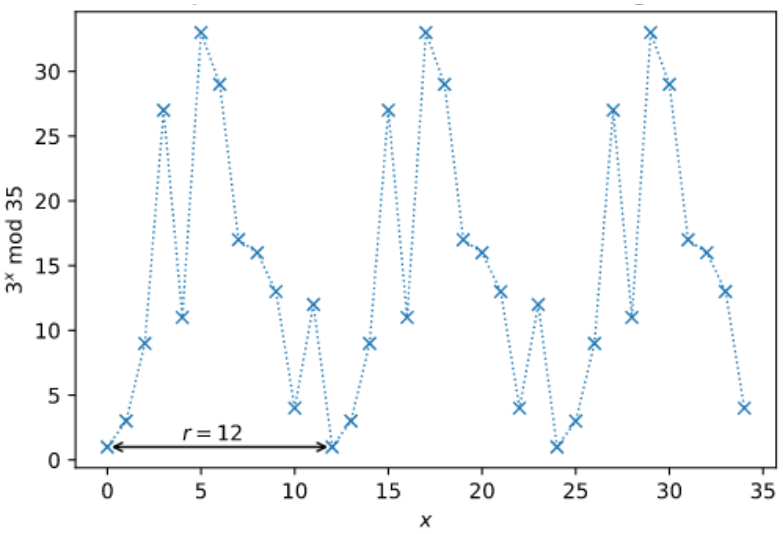
O período r da função, que é o menor número inteiro, sem ser zero, é dado pela equação

$$a^r \, mod \, N = 1 \qquad (5)$$

onde r é o período da função e a é um número escolhido

Na Figura 1 é mostrado o gráfico de uma função periódica no algoritmo de Shor, com N de 35, a igual a 3 e $r = 12$.

Figura 1 - Exemplo de uma função periódica no algoritmo de Shor.



Fonte: (QISKIT).

Utilizando a Estimativa de Fase Quântica, com o estado inicial em 1, temos que a cada aplicação consecutiva da função periódica, o estado será multiplicado por $a\ (mod\ N)$. Depois de r aplicações da função, o estado retornará à 1. Até esse ponto é possível usar um computador clássico para encontrar r. Dada a propriedade de superposição da computação quântica, é possível aplicar todas as funções periódicas de uma só vez. Entretanto este processo é limitado pela quantidade de *qubits* disponíveis no processador para registrar os estados. Para encontrar a frequência é utilizada a transformada quântica de Fourier. Para obter os fatores de N é necessário calcular o maior divisor comum de $a^{r/2} \pm 1$ em relação a N (IBM QUANTUM).

METODOLOGIA

Foi desenvolvido um programa para gerar as chaves do algoritmo RSA, e para, utilizando do algoritmo de Shor, encontrar os fatores primos da multiplicação. Deste modo, após encontrados, é gerada a chave privada a partir da chave pública. Assim é possível descriptografar a mensagem.

Para o desenvolvimento do programa que realiza o algoritmo RSA foi usado como base a implementação desenvolvida por CHEN (2017). A fim de melhorar o entendimento,

foram utilizados os conceitos do *clean architecture* (MARTIN, 2017). Deste modo, o código foi dividido em duas entidades e em casos de uso.

Para desenvolver a parte quântica, foi utilizado o SDK (Software Development Kit) Qiskit (QISKIT). O SDK do Qistik disponibiliza uma implementação do algoritmo de Shor, para ser executada nos simuladores ou nos computadores quânticos da IBM. Para o experimento foi utilizada a classe Shor do SDK, que recebe uma instância de um computador quântico, seja simulado ou real. Para executar é preciso chamar o método *factor* e passando o n como parâmetro junto de um inteiro a. O retorno do *factor* é um objeto que contém os fatores e outras informações de execução do algoritmo. Com isso, é possível usar os fatores para gerar a chave privada utilizando a chave pública (QISKIT).

RESULTADOS

O programa inicia atribuindo um valor inteiro à mensagem e em seguida são geradas as chaves para criptografar a mensagem. Depois é criado um *backend*, que nesse primeiro caso é realizado em uma máquina local, para conter a instância do simulador do computador quântico. Após, é instanciada a classe Shor, passando a instância e executado o *factor* sobre o produto. Assim, tendo os fatores é então descriptografada e exibida a mensagem. Isso é mostrado na Figura 2.

Figura 2 - Resultado da execução do programa.

Fonte: De autoria própria.

Os testes iniciais foram executados a partir do simulador executando em uma máquina local. Por esse motivo a escolha dos números primos de valor baixo, neste caso 3 e 7. Em seguida, utilizando o simulador nas máquinas da IBM foi possível simular com uma quantidade maior de *qbits*, permitindo usar números maiores. Deste modo, para os números primos 5 e 23 e n de 115, foram necessários 30 *qbits*, conforme mostrado na Figura 3.

Figura 3 - Resultado da execução do programa..

```
Prime p:  5
Prime q:  23
original d:  59
ciphertext:  39
n:  115
bit length:  7
qbits:  30
factors:  [5, 23]
generated d:  59
message:  9
```

Fonte: De autoria própria.

Os testes não puderam ser executados em computadores quânticos reais, devido a limitação de uso com de 5 qubits do processador disponibilizado no IBM Quantum de forma gratuita (IBM QUANTUM). Por exemplo, para encontrar os fatores do número 21 são necessários 22 *qubits*.

CONSIDERAÇÕES FINAIS

Conforme mostrado nos resultados, o programa desenvolvido foi capaz de quebrar a criptografia para números pequenos, dadas as limitações das simulações. Não foi possível executar o programa no computador quântico do IBM Quantum, dada a limitação do uso gratuito de 5 *qubits*. Como computadores quânticos não são triviais, a aplicação é bastante restrita.

Outro ponto importante a ser considerado é que as chaves RSA usadas em certificados SSL por exemplo, possuem entre 2048 e 4096 bits. Assim, utilizando o SDK da Qiskit para gerar o circuito onde calcula a quantidade de qubits necessários, uma chave de 4096 bits necessita 16386 *qubits*. Segundo SPARKES (2021), o computador quântico mais avançado é o da IBM, com 127 qubits. Isso demonstra que, utilizando esse método, ainda irá levar tempo para conseguir quebrar as chaves amplamente utilizadas.

REFERÊNCIAS BIBLIOGRÁFICAS

CHEN, Jane. RSA Implementation in Python, Abril 2017. Disponível em: <https://github.com/jchen2186/rsa-implementation>. Acesso em: 28 fev. 2022.

IBM QUANTUM. Disponível em:

<https://quantum-computing.ibm.com/composer/docs/iqx/guide/shors-algorithm>. Acesso em: 07 jun. 2022.

MILANOV, Evgeny. The RSA Algorithm. jun. 2009.

QUANTIKI. Disponível em: <https://www.quantiki.org/wiki/shors-factoring-algorithm>. Acesso em: 07 jun. 2022.

QISKIT. Disponível em: <https://qiskit.org/textbook/ch-algorithms/shor.html>. Acesso em: 07 jun. 2022.

SPARKES, Matthew. IBM creates largest ever superconducting quantum computer. Novembro 2021. Disponível em: <https://www.newscientist.com/article/2297583-ibm-creates-largest-ever-superconducting-quantum-computer/>. Acesso em: 01 mar. 2022.

MARTIN, Robert C.. Clean Architecture: A Craftsman's Guide to Software Structure and Design. 1ª Edição, 2017.